Using the Internet Gopher Protocol to Link a Computerized Patient Record and Distributed Electronic Resources

Joseph W. Hales, PhD, Division of Medical Informatics,
Richard C. Low, Medical Center Information Systems,
Kevin T. Fitzpatrick, PA-C, Medical Center Information Systems,
Duke University Medical Center, Durham, North Carolina

At Duke University Medical Center, we are developing a prototype clinical application for automated patient care plans with integrated links to electronic documents and other electronic resources. These links are implemented using the Internet Gopher Protocol, an emerging standard for distributed document search and retrieval. Use of this protocol permits storage of electronic documents in an open, nonproprietary manner. This paper discusses the architecture of the link mechanism and presents some of the advantages and disadvantages of the proposed method.

INTRODUCTION

One of the great advantages of automated clinical applications is the ability to link patient data to information about that data, in a hypertext fashion. These links can provide definitions, explanations, examples, reminders, and educational information. Futuristic depictions describe clinical information systems that link a diagnosis to an electronic text describing the diagnosis or to the results of an automatically executed bibliographic search of the latest therapy for the problem[1]. Further, one of the goals of the IAIMS effort is to "provide institution-wide access to appropriate bibliographic and knowledge databases" and present them in an integrated fashion[2].

A number of researchers have implemented or proposed models for supporting links between clinical applications and electronic resources such as online documentation and bibliographic databases [3,4,5,6,7,8,9,10].

Typically, attempts to link electronic documents or other resources to a clinical application have taken one of two approaches. The first approach is to tightly couple the document or external resource through some proprietary link or through wholly incorporating the resource into the application. For example, TMR is linked in this way to the Duke University Medical Center Laboratory Manual[11,12]. While reviewing patient laboratory results, the user can request to see the contents of the manual for a specific test. The text of the manual is coded in the same proprietary data structure used by TMR and requires a program written in the TMR application language (GEMISCH) to access the manual.

The second approach is to provide a simple interface between the clinical application to online resources as a stand alone program. This often fails to hide from the user differences in the interfaces of the different applications. A common example is using telnet to connect from a host running one application, such as a laboratory information system, to another host running a second application, such as a bibliographic retrieval application. For example, at Duke University Medical Center, a TMR-MEDLINE link passes previously determined search criteria to a search program searching the current literature for information on patient specific acid-base imbalances[3].

Ideally, there is a balance between highly integrated resources that give the impression one application and stand alone resources that can be accessed independently. At the same time, the links between clinical applications and electronic resources should be context sensitive. Based on the patient's condition or the context of information presently on the computer screen, the links should point to the most related (useful) part of the electronic resource. Recently, standards have emerged that support these seemingly contradictory goals.

GOPHER

The Internet Gopher Protocol[13] is an emerging standard for distributed document search and retrieval on TCP/IP networks. Documents may be ASCII text, images, sound or even executable program files. The protocol is based on a client-server model. Documents reside on autonomous servers throughout the Internet. Client software connects to a server and submits a request; the server responds with a block of data

0195-4210/92/$5.00

interpreted by the client and terminates the connection. The server's response may be a document, a list of documents or a pointer to some service (e.g., search service or telnet). The Gopher client displays the returned document or presents the list of documents as hierarchical menu, like a file system directory structure. The hierarchical directory paradigm is dynamically constructed and hides from the user the fact that the documents may reside on different servers on different host platforms.

The server's response to a client's initial request provides the information necessary for unique document requests by the client. The server's response is composed of strings of text, one line for each document or list of documents available from the server. An example of a server response is shown in Figure 1. The format of each line is a series of fields delimited by tabs (indicated by → in Figure 1). The first character of the first field represents the document type; the remaining characters in the first field are the name of the item to be displayed by the client to the user. The document type directs the client how to interpret the result of the request (e.g., display a document or display a menu of documents). The second field represents the "selector" string that should be sent to the remote host to request the item listed. The remaining two fields represent the IP domain name of the server host and the port at which to connect to request this item. The client software can locate and retrieve any item from a server by a tuple of selector, host name, and port. Subsequent queries by the client are simply transmissions of the selector string to the specified host at the specified port.

The protocol is connectionless. All necessary information for subsequent queries is passed to the client. As a result, the client need not maintain elaborate tables of linkage information, since this is obtained from the server as part of the connection. The protocol does not preclude, however, the client from maintaining its own table of selector tuples.

By adding the document type to the tuple of "selector," host name, and port, a document on a gopher server is uniquely identified. These tuples can be saved for direct access without initialization. The simplicity of the Gopher protocol and the unique description of a gopher server document determined by the four-part tuple permit a client application to be written that communicates with servers as a regular client. However, this customized client can retrieve specific documents buried deep in the directory structure, yet mask the full hierarchy of documents on the server from the user.

PROTOTYPE APPLICATION

At Duke University Medical Center, we are prototyping a customized Gopher client imbedded as part of a developmental Care Map application. Care Maps are being developed and implemented as multi-disciplinary guidelines for patient care. Care Maps describe 1) common problems; 2) expected patient outcomes of those problems; and 3) the multi-disciplinary interventions that are made to achieve those outcomes for 75% of the patients with the problem.

Care Maps are an important component of the Clinical Workstation. Using a spreadsheet-like display, the Care Map serves as a multi-dimensional entry point into a computerized patient record. One desirable aspect of an automated Care Map application is links to electronic resources, such as online documentation and clinical event annotations. For this capability we turned to the Internet Gopher Protocol.

The Care Map imbedded Gopher integrates the Internet Gopher Protocol and portions of the Gopher client software for communication with the server and document presentation. The resulting application will smoothly integrate the retrieval of distributed documents into the Care Map application. The Care Map application is being developed on the NeXT (NeXT, Inc., Redwood

```
1Information about Gopher→1/Information about Gopher→nameserver2.mc.duke.edu→70
1Medical Center Information→1/Medical Center Information→nameserver2.mc.duke.edu→70
1NIH Guide to Grants and Contracts→1/nih/nihguide→helix.nih.gov→70
1Biomedical related Information Resources→1/bio→camis.Stanford.EDU→70
1Sampler→1/Sampler→nameserver2.mc.duke.edu→70
8Triangle Research Libraries→→LIBROT1.LIB.UNC.EDU→23
0Weather Forecast for Raleigh Durham→0/Weather/North Carolina/Raleigh-Durham→ashpool.micro.umn.edu→70
.
```

Figure 1 An example of a response from a gopher server. Each line represents a document or list of documents. Fields are separated by tabs, represented here by →.

City, CA) computer running NeXTSTEP 3.0. The Care Map application and imbedded gopher are built using NeXT's Interface Builder application and custom Objective C code. The Care Map database is maintained in a Sybase (Sybase, Inc., Emeryville, CA) relational database.

Objects (Care Map events) in the Care Map application that may require elaboration or explanation are linked to a reference table in the care map database schema. This table of references lists one or more selector tuples for "Gopher documents" that are appropriately related to the even. (An example of the schema is shown in Table 1.) Selecting the link option for an object will cause selector tuples to be submitted to the customized Gopher client. If a single reference exists, the item (document directory) will be requested directly. If multiple references exist, the client will display these as a dynamically constructed Gopher menu. The imbedded Gopher client will display the results of a request or dynamic menu in a pop up window as shown in Figure 2.

As of this writing, the imbedded Gopher client can communicate with servers, but the reference list has not yet been implemented. We are presently building texts on a Gopher server running on a Sun 4 (Sun Microsystems Computer Corp, CA) workstation.

The following is a scenario describing the links in the Care Map prototype using the Internet Gopher Protocol. Operation of the full Care Map application is described elsewhere[14]. A user, while reviewing laboratory results, sees an alert regarding an abnormal serum foobar value which is suggestive of acute hypertension. The user wants to consult the links for further information. Three links are stored in the reference table. The client presents these as three menu choices. The first link points to a paper on the Duke Gopher server showing hypertension as a risk factor for intracranial Hemorrhage. The second reference is a link to the medical logic module (MLM) that evoked the alert. This module resides on the prototype MLM Gopher server at Columbia Presbyterian Medical Center. The third reference points at a binary image file showing a 3-D image of the foobar molecule. This image is located at the University of Utah on the fictitious Slice of Life Gopher server.

Table 1 Elements of reference link pointer database schema.

Column Name	Type	Size
CareMapEvent_id	int	4
type	char	1
path	varchar	255
host	varchar	255
port	int	4

DISCUSSION

The scenario above is different from many described elsewhere in the literature, perhaps only because the electronic links are distributed over the Internet. The integration of distributed electronic resources is not revolutionary. However, the use of the Internet Gopher Protocol adds a standard for retrieval of distributed documents and a method for integrating the retrieval engine into clinical applications in a more seamless fashion than previously achieved.

Advantages

There are several advantages to the model we have described. We have already noted what we believe to be a new degree seamless integration. At the same time, the protocol is an entirely open method of distributing online documentation. This "openness" is the foremost advantage of use of the Gopher protocol in this application. A standard protocol permits existing Gopher clients or other applications that support the protocol to communicate with the server established to service care map link requests. A properly constructed document on a gopher server can be presented as a complete, organized document to existing client software and at the same time support requests from applications like the imbedded care map gopher for specific documents or subtrees of documents buried in the hierarchy of the document structure. Additionally, the client server design permits client applications to be written for any platform that supports TCP/IP.

The protocol's support for distributed documents permits the same resource to be shared by multiple institutions without necessarily distributing copies. As suggested by the scenario above, a useful collection of related documents can be presented to the user as a single list of documents, despite the fact that the documents are widely distributed. This distribution potentially

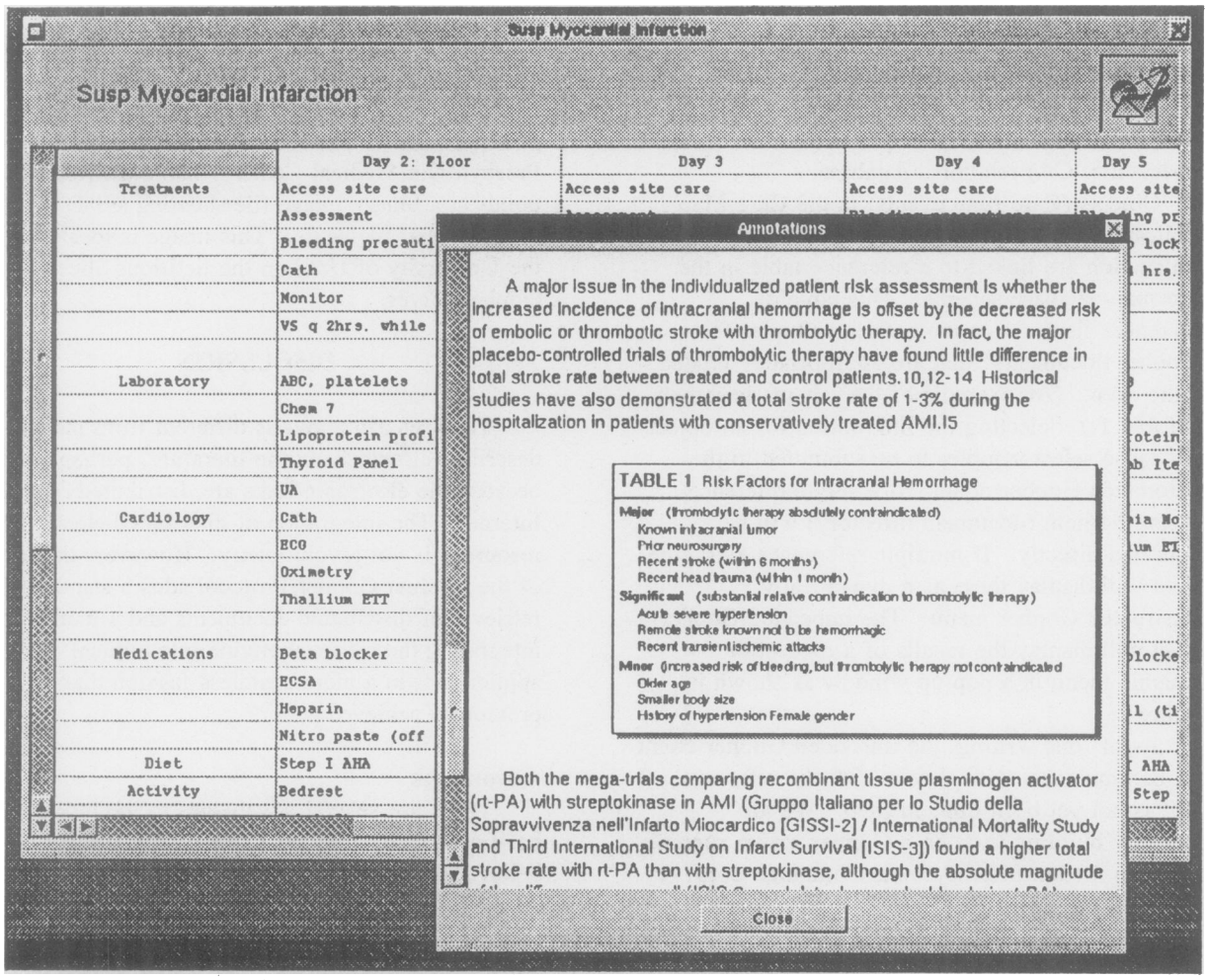

Figure 2 Example of how a window of linked text retrieved using an imbedded gopher client will appear.

reduces maintenance overhead, since original documents can be managed "local" to the document producer. Ideally, institutions could identify themselves as domain or specialty repositories and serve as the primary developer and maintainer of electronic documents in those domains.

Disadvantages

Most disadvantages of using the Internet Gopher Protocol to support electronic links center around the minimalist nature of the protocol. While a variety of documents can be retrieved, including text and images, many formats are not supported and the documents are not interactive. Gopher does not support hypertext linkages within or between retrieved documents. Additionally, in practice, network access may not turn out to be sufficiently reliable for geographically distant requests, lacking speed or consistency. Performance could be improved by redundant distribution of servers.

Finally, maintenance of the reference pointers within the application may be difficult. The Gopher Protocol was developed to support an interactive client. At initialization, interactive Gopher clients receive from the server all of the information necessary for subsequent requests. Since the customized client using a stored reference table does not request the initial structure of the server, changes in the document structure of the server are not reflected nor updated in the reference table. This updating is less important if the documents are relatively static. Additionally, by pointing at subtrees of the document structure rather than individual documents, the impact of some changes can be reduced.

CONCLUSION

Using the Internet Gopher Protocol, an emerging standard for distributed document retrieval, we have proposed a model for integrating

networked electronic resources with clinical applications. This model supports a high level of integration while still retaining open access to the electronic resources. Additionally, we have built a prototype system implementing portions of this model.

Proposal

In conclusion we offer a proposal to the medical informatics community. We recommend a formal effort to coordinate the development of medical informatics related Gopher servers. We believe friendly cooperation between institutions may reduce redundancy through, for example, establishment of dedicated domain-specific servers like the prototype MLM server at Columbia University. Additionally, a coordinated effort can develop guidelines for application links to distributed texts accessible via the Gopher protocol. Finally, a community effort will likely be needed to establish mechanisms to address the issues of intellectual property of electronic biomedical documents.

ACKNOWLEDGMENTS

This work is supported in part by National Library of Medicine Grant G08-LM-0-4613-06.

Reference

[1]. Shortliffe EH, Perrault LE, Wiederhold G, Fagan LM (eds.). Medical Informatics, Computer Applications in Health Care. Reading, Massachusetts: Addison-Wesley Publishing Co.; 1990.

[2]. Stead WW, et al. IAIMS-the role of strategic planning. Proceedings of the 13th Annual SCAMC. 1989; 13:345-349.

[3]. Hammond JE, Hammond WE, Stead WW. Information Management Through Integration of Distributed Resources: The TMR-NLM Connection as a Prototype. Proceedings of the 14th Annual SCAMC. 1990;14:719-723.

[4]. Frisse ME, Cousins SB. Query by Browsing: An Alternative Hypertext Information Retrieval Method. Proceedings of the 13th Annual Symposium on Computer Applications in Health Care. 1989;13:388-391.

[5]. Cimino C, Barnett GO. Standardizing Access to Computer-Based Medical Resources. Proceedings of the 14th Annual SCAMC. 1990;33-37.

[6]. Deibel SRA, Greenes RA, Snydr-Michal JT. DeSyGNER: A Building Block Architecture Fostering Independent Cooperative Development of Multimedia Knowledge Management Applications. Proceedings of the 14th Annual SCAMC. 1990;14:445-449.

[7]. Clark AS, Shea S. Free Text Databases in an Integrated Academic Information System (IAIMS) at Columbia Presbyterian Medical Center. Proceedings of the 15th Annual SCAMC. 1991;15:333-337.

[8]. Loonsk, JW, Lively R, TinHan E, Litt H. Implementing the Medical Desktop: Tools for the Integration of Independent Information Resources. Proceedings of the 15th Annual SCAMC. 1991;15:574-577.

[9]. Cimino JJ, Johnson SB, Aguirre A, Roderer N, Clayton PD. The Medline Button. Proceedings of the 16th Annual SCAMC. 1992;16:81-85.

[10]. Hales JW, Gardner RM, Huff SM. Integration of a Stand-alone Expert System with a Hospital Information System. Proceedings of the 16th Annual SCAMC. 1992;16:427-431.

[11]. Stead WW, Hammond WE. Computer-Based Medical Records: the Centerpiece of TMR. MD Computing. 1988;5(5):48-62.

[12]. Grewal R, Arcus J, Bowen J, Fitzpatrick KT, Hammond WE, Hickey L, Stead WW. Bedside Computerization of the ICU, Design Issues: Benefits of Computerization Versus Ease of Paper & Pen. Proceedings of the 15th Annual SCAMC. 1991;15:793-797.

[13]. Anklesaria F, et al. The Internet Gopher Protocol. Network Working Group Request for Comments: 1436; March 1993.

[14]. Fitzpatrick KP, Low RC, Campbell J, Boyarsky W, Pickett MP. An Object Oriented Clinical Care Map Application Utilizing the NeXTSTEP™ Operating System. Submitted to Seventeenth Annual SCAMC.